CBCS SCHEME

USN 1CR22KI022

Sixth Semester B.E./B.Tech. Degree Examination, June/July 2025
Machine Learning

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	State Tom Mitchell's definition of machine learning. List and explain the challenges of machine learning.	7	L1	CO1
	b.	List and explain the visualization aids available for univariate data analysis with example for each.	7	L2	CO1
	c.	For the patients age list {12, 14, 19, 22, 24, 26, 28, 31, 34}. Find the IQR.	6	L3	CO1
		OR			
Q.2	a.	Explain in detail the machine learning process with a neat diagram.	7	L2	CO1
	b.	Explain data preprocessing with measures to solve the problem of missing data.	7	L2	CO1
	c.	Find the 5-point summary of the list {13, 11, 2, 3, 4, 8, 9} and plot the box plot for the same.	6	L3	CO1
		Module – 2			
Q.3	a.	Let the data points be $\begin{pmatrix}2\\6\end{pmatrix}$ and $\begin{pmatrix}1\\7\end{pmatrix}$. Apply Principal Component Analysis (PCA) and find the transformed data.	10	L3	CO1
	b.	Apply candidate elimination algorithm on the dataset given in Table Q.3(b) to obtain the complete version space.	10	L3	CO2

Table Q.3(b)

CGPA	Interactiveness	Practical knowledge	Communication skills	Logical thinking	Job offer
≥9	Yes	Excellent	Good	Fast	YES
≥9	Yes	Good	Good	Fast	YES
≥8	No	Good	Good	Fast	NO
≥9	Yes	Good	Good	Slow	YES

		OR			
Q.4	a.	Find Singular Value Decomposition (SVD) of the matrix $A = \begin{pmatrix}1 & 2\\4 & 9\end{pmatrix}$.	10	L3	CO2

			M	L	C
	b.	Write Find-S algorithm. Apply the algorithm to obtain the hypothesis for the dataset given in the Table Q.4(b).	10	L3	CO2

Table Q.4(b)

Sky	Air temp	Humidity	Wind	Water	Forecast	Enjoy sport
Sunny	Warm	Normal	Strong	Warm	Same	YES
Sunny	Warm	High	Strong	Warm	Same	YES
Rainy	Cold	High	Strong	Warm	Change	NO
Sunny	Warm	High	Strong	Cool	Change	YES

		Module – 3			
Q.5	a.	Apply K-nearest neighbor algorithm for the dataset given in Table Q.5(a). Given a test instance (6.1, 40, 5) use the training set to classify the test instance. Choose K = 3	6	L3	CO3

Table Q.5(a)

CGPA	Assessment	Project submitted	Result
9.2	85	8	PASS
8	80	7	PASS
8.5	81	8	PASS
6	45	5	FAIL
6.5	50	4	FAIL
5.8	38	5	FAIL

			M	L	C
	b.	Explain types of regression methods and limitations of regression methods.	7	L2	CO3
	c.	Explain the structure of a decision tree and write the procedure to construct a decision tree using ID3 algorithm.	7	L2	CO3
		OR			
Q.6	a.	Write the nearest-centroid classifier algorithm. Apply the same to predict the class for the given test instance (6, 5) using the training dataset given in Table Q.6(a).	7	L3	CO3

X	Y	Class
3	1	A
5	2	A
4	3	A
7	6	B
6	7	B
8	5	B

Table Q.6(a)

			M	L	C
	b.	Distinguish between i) Regression and correlation ii) Regression and causation iii) Linearity and non-linearity relationships	6	L2	CO3
	c.	Explain the advantages and disadvantages of decision tree. Write the general algorithm for decision tree.	7	L2	CO3

		Module – 4	M	L	C
Q.7	a.	Using Naïve bayes classifier classify the new data (Red, SUV, Domestic) using the training dataset given in Table Q.7(a).	10	L3	CO4

Table Q.7(a)

Color	Type	Origin	Stolen
Red	Sports	Domestic	YES
Red	Sports	Domestic	NO
Red	Sports	Domestic	YES
Yellow	Sports	Domestic	NO
Yellow	Sports	Imported	YES
Yellow	SUV	Imported	NO
Yellow	SUV	Imported	YES
Yellow	SUV	Domestic	NO
Red	SUV	Imported	NO
Red	Sports	Imported	YES

			M	L	C
	b.	Explain the simple model of an artificial neuron along with the artificial neural network structure.	10	L2	CO4
		OR			
Q.8	a.	Explain Bayes theorem, Maximum A Posteriori (MAP) hypothesis and Maximum Likelihood (ML) hypothesis in detail.	10	L2	CO4
	b.	Explain different activation functions used in artificial neural network.	10	L2	CO4
		Module – 5			
Q.9	a.	Consider the following set of data given in Table Q.9(a). Cluster it using K-means algorithm with initial value of objects 2 and 5 with the coordinate values (4, 6) and (12, 4) as initial seeds.	10	L3	CO5

Table Q.9(a)

Objects	X-coordinate	Y-coordinate
1	2	4
2	4	6
3	6	8
4	10	4
5	12	4

			M	L	C
	b.	Explain the various components of reinforcement learning.	10	L2	CO5
		OR			
Q.10	a.	Find the Manhattan and Chebyshev distance if the coordinates of the objects are (0, 3) and (5, 8).	4	L3	CO5
	b.	Explain the mean shift clustering algorithm.	6	L2	CO5
	c.	List and explain the: i) Characteristics of reinforcement learning ii) Challenges of reinforcement learning iii) Applications of reinforcement learning	10	L3	CO5

* * * * *

BCS602-MACHINE LEARNING-1

ANSWER KEY

1a) According to **Tom M. Mitchell**, a well-known ML researcher, the formal definition of machine learning is:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."
— *Tom M. Mitchell, "Machine Learning", McGraw Hill, 1997*

Explanation:

- **Task (T):** What the system is trying to do (e.g., classify emails as spam or not spam).
- **Experience (E):** The data or interactions the system uses to learn (e.g., past labeled emails).
- **Performance (P):** How well the system performs (e.g., classification accuracy).

Challenges of Machine Learning

1. **Insufficient Quantity of Training Data**
   - ML models require large amounts of labeled data to perform well.
   - Small datasets lead to **underfitting**, poor generalization, and high error rates.
2. **Noisy and Incomplete Data**
   - Real-world data often contains errors (noise), missing values, or inconsistent entries.
   - Noise can confuse the learning algorithm and affect model accuracy.
3. **Overfitting and Underfitting**
   - **Overfitting:** Model memorizes training data, fails on unseen data.

- **Underfitting:** Model is too simple to learn underlying patterns.
- Proper regularization and validation are required.

4. **High Dimensionality**
    - Too many input features (dimensions) can make learning inefficient or inaccurate.
    - This is called the **"curse of dimensionality"**.
    - Feature selection and dimensionality reduction (like PCA) help alleviate this.
5. **Imbalanced Data**
    - Occurs when one class dominates (e.g., 95% class A, 5% class B).
    - The model may get biased toward the majority class, ignoring minority class performance.
6. **Interpretability and Explainability**
    - Complex models (e.g., deep neural networks) are hard to interpret.
    - In critical domains like healthcare or finance, explainable AI is essential.
7. **Scalability**
    - ML algorithms should handle large datasets efficiently.
    - Issues like memory usage and time complexity become bottlenecks with big data.
8. **Data Privacy and Security**
    - Learning from sensitive data (e.g., medical records) poses privacy concerns.
    - Techniques like **differential privacy** and **federated learning** are used to protect data.
9. **Concept Drift**
    - In dynamic environments, the statistical properties of the target variable change over time.

    - Example: user preferences on a streaming platform.
    - The model must adapt continuously.

10. Bias and Fairness

- ML systems may learn and perpetuate societal biases present in training data.
- Fairness-aware learning is crucial to ensure equitable outcomes.

1b) **Univariate Data Visualization Aids**

Univariate analysis involves examining **one variable** at a time. Visualization helps understand the **distribution, central tendency, spread, and outliers**.

1. Histogram

- **Definition:** A bar chart representing the **frequency distribution** of a single continuous variable.
- **Usage:** Shows how data is distributed over intervals (bins).
- **Insight:** Shape of distribution (normal, skewed, bimodal), spread, central value.

Example: Histogram of students' marks to see how many scored within certain ranges.

---

2. Box Plot (Box-and-Whisker Plot)

- **Definition:** Displays **median, quartiles, minimum, maximum**, and **outliers**.
- **Usage:** Useful for comparing distributions or detecting skewness and outliers.
- **Insight:** Highlights **spread**, **central tendency**, and **extreme values**.

Example: Box plot of income levels to show spread and outliers.

---

3. Bar Chart

- **Definition:** Represents **categorical univariate data** using rectangular bars.
- **Usage:** Displays **frequency or proportion** of categories.
- **Insight:** Helps compare different categories.

Example: Bar chart showing number of students in different majors (CS, ECE, ME).

---

4. Pie Chart

- **Definition:** A circular chart divided into **slices** to show **proportions**.
- **Usage:** Best for showing **percentage share** of categories.
- **Insight:** Understand composition of categorical data.

Example: Pie chart of browser usage among users.

---

5. Line Plot

- **Definition:** Graph that uses points connected by lines to show data over **time or order**.
- **Usage:** Best for ordered univariate data or time series.
- **Insight:** Trends, patterns, or fluctuations over time.

Example: Line plot of daily temperatures in a week.

---

6. Stem-and-Leaf Plot

- **Definition:** Text-based plot where numbers are split into **stem (leading digit)** and **leaf (trailing digit)**.
- **Usage:** Good for small datasets.
- **Insight:** Retains original data values, shows distribution and shape.

Example: Stem-and-leaf plot of test scores like 78, 82, 85.

---

7. Frequency Table

- **Definition:** A tabular representation of **value counts**.
- **Usage:** Aids in creating histograms or bar charts.
- **Insight:** Summary of how often each value occurs.

Example: Table showing frequency of rainfall levels (0–10 mm, 10–20 mm, etc.)

1c)

Step 1: Find **Q1** (First Quartile)

Q1 is the **25th percentile**, or the **median of the lower half** (excluding the median if n is odd).

Lower half = `{12, 14, 19, 22}`

Median of lower half (Q1) = average of 14 and 19 =

$$Q1 = \frac{14 + 19}{2} = \frac{33}{2} = 16.5$$

Step 2: Find **Q3** (Third Quartile)

Q3 is the **75th percentile**, or the **median of the upper half** (excluding the median if n is odd).

Upper half = `{26, 28, 31, 34}`

Median of upper half (Q3) = average of 28 and 31 =

$$Q3 = \frac{28 + 31}{2} = \frac{59}{2} = 29.5$$

Step 3: Calculate **IQR**

$$\text{IQR} = Q3 - Q1 = 29.5 - 16.5 = \boxed{13}$$

2a) **1. Problem Definition**

- Clearly define the **goal** of the ML system.
- Understand what is to be predicted or classified.
- Define input features and output (target) variable.

📌 *Example:* Predict house prices based on features like size, location, and number of rooms.

---

2. Data Collection

- Gather relevant data from various sources (e.g., sensors, logs, databases, surveys).
- Data can be **structured** (tabular) or **unstructured** (text, images).

⚠️ *Bad or insufficient data = poor model performance.*

---

3. Data Preprocessing

- Clean and prepare the data for training.
- Steps may include:
  - Handling missing values
  - Removing duplicates
  - Encoding categorical variables
  - Normalization or standardization
  - Outlier detection and removal

💡 *"Garbage in, garbage out" — clean data is essential for accurate models.*

---

4. Feature Selection and Extraction

- Select the most relevant input features that influence output.
- Feature engineering: derive new features using domain knowledge.
- Dimensionality reduction (e.g., PCA) may be applied to reduce redundancy.

🧮 *Better features = better learning = better results.*

---

5. Splitting the Dataset

- Divide data into:
    - **Training set**: to train the model
    - **Validation set**: to tune parameters (optional)
    - **Test set**: to evaluate final performance

⚖️ *Typical split: 70% training, 15% validation, 15% testing*

---

6. Model Selection

- Choose an appropriate learning algorithm based on the problem type:
    - **Classification** (e.g., decision trees, SVM)
    - **Regression** (e.g., linear regression)
    - **Clustering** (e.g., K-means)

🧠 *Understanding the nature of data and target helps choose the right model.*

---

7. Model Training

- Feed training data into the algorithm to allow it to learn patterns.
- Parameters are adjusted to minimize the loss/error function.

🔁 *May require multiple iterations (epochs) to optimize performance.*

---

8. Model Evaluation

- Evaluate the model using metrics like:
    - Accuracy, Precision, Recall, F1-score (for classification)
    - RMSE, MAE, R^2 (for regression)

📊 *Evaluation helps understand generalization to unseen data.*

---

9. Model Tuning / Hyperparameter Optimization

- Tune model hyperparameters using techniques like:

  - Grid Search
  - Random Search
  - Cross-validation

🔧 *Example: adjusting learning rate, number of trees in Random Forest.*

---

10. Deployment

- Deploy the trained model in a real-world environment (e.g., web app, embedded system).
- Monitor performance and update as needed.

🚀 *Deployment brings your ML model into production use.*

---

11. Monitoring and Maintenance

- Continuously monitor model performance.
- Handle **concept drift** (changes in data patterns).
- Retrain model periodically with fresh data.

+diagram

2b) **Data Preprocessing Measures to Handle Missing Values**

1. Ignore the Tuple (Row Deletion)

- Remove records (rows) that contain missing values.
- Works well **only when the number of such tuples is small**.

⚠️ **Not recommended** when a large portion of data is missing.

---

2. Fill in Manually

- Missing values are **filled manually by experts** based on domain knowledge.
- Accurate but **not scalable** for large datasets.

📌 *Useful for small datasets or critical fields like medical data.*

---

3. Use a Global Constant

- Replace all missing values with a constant (e.g., “Unknown” or -9999).
- Useful for **categorical data**.

⚠️ May **distort data distribution** or bias some models.

---

4. Use Attribute Mean/Median/Mode

- **Numerical Attributes:**
    - Replace missing values with **mean** or **median**.
- **Categorical Attributes:**
    - Replace with the **mode** (most frequent value).

✅ Simple and fast; widely used in practice.

⚠️ Mean is sensitive to outliers — use **median** if data is skewed.

---

5. Use Class-Specific Mean/Median/Mode

- If target class is known, fill missing values using mean/median/mode **of that class only**.

📌 *More accurate than global average — preserves class-based distinctions.*

---

6. Predict Missing Value Using a Model

- Build a **classification or regression model** to predict the missing value using other attributes.
    - Use K-Nearest Neighbors (KNN), Decision Trees, etc.
- Considered a **smart imputation** method.

✅ Effective for complex patterns, but computationally expensive.

---

7. Use Data Imputation Techniques

- Advanced statistical techniques:
  - **KNN imputation**
  - **Multiple Imputation**
  - **Expectation-Maximization (EM)** algorithm

💡 These methods preserve statistical relationships between attributes.

---

8. Use Interpolation (for Time Series Data)

- Estimate missing values using interpolation methods (linear, polynomial) based on **time** or **sequence**.

📈 Very useful in time-series datasets (e.g., temperature logs, ECG data).

2c)

1. **Minimum (Min):**
   → Smallest value = 2
2. **First Quartile (Q1):**
   → Median of lower half: {2, 3, 4}
   → Q1 = 3
3. **Median (Q2):**
   → Middle value = 8
4. **Third Quartile (Q3):**
   → Median of upper half: {9, 11, 13}
   → Q3 = 11
5. **Maximum (Max):**
   → Largest value = 13

+Plot the box plot

3a)

Step 1: Mean Center the Data

Compute column-wise mean:

- Mean of column 1: $\frac{2+6}{2} = 4$
- Mean of column 2: $\frac{1+7}{2} = 4$

Subtract the mean:

$$A_{\text{centered}} = A - \text{mean} = \begin{bmatrix} 2-4 & 1-4 \\ 6-4 & 7-4 \end{bmatrix} = \begin{bmatrix} -2 & -3 \\ 2 & 3 \end{bmatrix}$$

Step 2: Compute Covariance Matrix

$$\text{Cov} = \frac{1}{n-1} A_{\text{centered}}^T A_{\text{centered}}$$

$$A_{\text{centered}}^T = \begin{bmatrix} -2 & 2 \\ -3 & 3 \end{bmatrix} \Rightarrow \text{Cov} = \frac{1}{1} \begin{bmatrix} (-2)^2 + 2^2 & (-2)(-3) + 2(3) \\ (-3)(-2) + 3(2) & (-3)^2 + 3^2 \end{bmatrix} = \begin{bmatrix} 8 & 12 \\ 12 & 18 \end{bmatrix}$$

Step 3: Compute Eigenvalues and Eigenvectors

Let's solve $\det(C - \lambda I) = 0$

$$\det \begin{bmatrix} 8-\lambda & 12 \\ 12 & 18-\lambda \end{bmatrix} = (8-\lambda)(18-\lambda) - 144 = \lambda^2 - 26\lambda + 0 = 0 \Rightarrow \lambda = 0,\ 26$$

Step 4: Eigenvectors

- For λ = 26:

$$(C - 26I)v = 0 \Rightarrow \begin{bmatrix} -18 & 12 \\ 12 & -8 \end{bmatrix} v = 0$$

Solve:

$$-18x + 12y = 0 \Rightarrow y = \frac{3}{2}x$$

→ Eigenvector: $v_1 = \begin{bmatrix} 1 \\ \frac{3}{2} \end{bmatrix}$

→ Normalize:

$$\text{norm} = \sqrt{1^2 + \left(\frac{3}{2}\right)^2} = \sqrt{1 + \frac{9}{4}} = \sqrt{\frac{13}{4}} = \frac{\sqrt{13}}{2}$$

$$\text{Normalized } v_1 = \frac{1}{\frac{\sqrt{13}}{2}} \begin{bmatrix} 1 \\ \frac{3}{2} \end{bmatrix} = \frac{2}{\sqrt{13}} \begin{bmatrix} 1 \\ \frac{3}{2} \end{bmatrix}$$

3b)

S_0 = [Ø, Ø, Ø, Ø, Ø] (most specific)

G_0 = [?, ?, ?, ?, ?] (most general)

Example 1 (Positive)

`[>=9, Yes, Excellent, Good, Fast] → Yes`

- S_1 ← Replace Ø with the example:

$$S_1 = [>= 9, Yes, Excellent, Good, Fast]$$

- **G_1 remains unchanged** (no need to generalize from positive):

$$G_1 = [?, ?, ?, ?, ?]$$

Example 2 (Positive)

`[>=9, Yes, Good, Good, Fast] → Yes`

- Compare with S_1:

```yaml
S₁:   [>=9, Yes, Excellent, Good, Fast]
New:  [>=9, Yes, Good,      Good, Fast]
```

- **Change "Excellent" to "?"** (since now both "Excellent" and "Good" are acceptable):

$$S_2 = [>= 9, Yes, ?, Good, Fast]$$

- G remains unchanged:

$$G_2 = [?, ?, ?, ?, ?]$$

Example 3

$$G_3 = \{[>= 9, ?, ?, ?, ?],\ [?, Yes, ?, ?, ?]\}$$

Example 4 (Positive)

`[>=9, Yes, Good, Good, Slow] → Yes`

S_2 = `[>=9, Yes, ?, Good, Fast]`

New = `[>=9, Yes, Good, Good, Slow]`

- Only difference is **Logical Thinking**: "Fast" vs "Slow"
  → Generalize it to `?`

✅ New S becomes:

$$S_4 = [>= 9, Yes, ?, Good, ?]$$

Now check G_3 again:

We remove any hypotheses that are **more general than S but inconsistent with new positive example**

- `[>=9, ?, ?, ?, ?]` matches → OK
- `[?, Yes, ?, ?, ?]` matches → OK

So,

✅ Final G:

$$G_4 = \{[>= 9, ?, ?, ?, ?],\ [?, Yes, ?, ?, ?]\}$$

S = [>=9, Yes, ?, Good, ?]

G = {[>=9, ?, ?, ?, ?], [?, Yes, ?, ?, ?]}

4a)

Step 1: Compute $A^T A$

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 9 \end{bmatrix} \Rightarrow A^T A = \begin{bmatrix} 1 & 4 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 4 & 9 \end{bmatrix} = \begin{bmatrix} 1^2 + 4^2 & 1 \times 2 + 4 \times 9 \\ 2 \times 1 + 9 \times 4 & 2^2 + 9^2 \end{bmatrix} = \begin{bmatrix} 17 & 38 \\ 38 & 85 \end{bmatrix}$$

Step 2: Compute Eigenvalues of $A^T A$

Find the eigenvalues λ from:

$$\det(A^T A - \lambda I) = 0$$

$$\begin{vmatrix} 17 - \lambda & 38 \\ 38 & 85 - \lambda \end{vmatrix} = (17 - \lambda)(85 - \lambda) - 38^2 = 0$$

Compute:

$$(17 - \lambda)(85 - \lambda) = \lambda^2 - 102\lambda + 1445$$
$$\Rightarrow \lambda^2 - 102\lambda + 1445 - 1444 = \lambda^2 - 102\lambda + 1 = 0$$

So,

$$\lambda = \frac{102 \pm \sqrt{102^2 - 4(1)(1)}}{2} = \frac{102 \pm \sqrt{10400}}{2} = \frac{102 \pm 102}{2} \Rightarrow \lambda_1 = 101, \lambda_2 = 1$$

Step 3: Singular Values (σ)

$$\sigma_i = \sqrt{\lambda_i} \Rightarrow \sigma_1 = \sqrt{101}, \ \sigma_2 = \sqrt{1} = 1$$

So,

$$\Sigma = \begin{bmatrix} \sqrt{101} & 0 \\ 0 & 1 \end{bmatrix}$$

Step 4: Compute V (Right Singular Vectors)

Solve $(A^T A)v = \lambda v$

You can compute the eigenvectors for λ = 101 and λ = 1 to get V, then normalize them.

Step 5: Compute U (Left Singular Vectors)

$$u_i = \frac{1}{\sigma_i} A v_i$$

Let:

$$A = \begin{bmatrix} 1 & 2 \\ 4 & 9 \end{bmatrix} \Rightarrow A = U\Sigma V^T$$

Where:

- $\Sigma = \begin{bmatrix} \sqrt{101} & 0 \\ 0 & 1 \end{bmatrix}$
- V: eigenvectors of $A^T A$
- U: calculated as Av_i/σ_i

4b)

Example 1:

[Sunny, Warm, Normal, Strong, Warm, Same] → Yes

S_1 = [Sunny, Warm, Normal, Strong, Warm, Same]

Example 2:

[Sunny, Warm, High, Strong, Warm, Same] → Yes

Compare with S_1:

- Normal vs High → conflict → generalize to ?
- Rest are same

S_2 = [Sunny, Warm, ?, Strong, Warm, Same]

Example 3:

[Rainy, Cold, High, Strong, Warm, Change] → No

→ Ignore negative example when computing S

Example 4:

[Sunny, Warm, High, Strong, Cool, Change] → Yes

Compare with S_2:

- Water: Warm vs Cool → conflict → generalize to ?
- Forecast: Same vs Change → conflict → generalize to ?

S_3 = [Sunny, Warm, ?, Strong, ?, ?]

5a)

Distance from (9.2, 85, 8) → Pass

$$\sqrt{(6.1-9.2)^2+(40-85)^2+(5-8)^2}$$
$$=\sqrt{9.61+2025+9}=\sqrt{2043.61}\approx 45.21$$

Distance from (8, 80, 7) → Pass

$$=\sqrt{(6.1-8)^2+(40-80)^2+(5-7)^2}$$
$$=\sqrt{3.61+1600+4}=\sqrt{1607.61}\approx 40.08$$

Distance from (8.5, 81, 8) → Pass

$$=\sqrt{(6.1-8.5)^2+(40-81)^2+(5-8)^2}$$
$$=\sqrt{5.76+1681+9}=\sqrt{1695.76}\approx 41.18$$

Distance from (6, 45, 5) → Fail

$$=\sqrt{(6.1-6)^2+(40-45)^2+(5-5)^2}$$
$$=\sqrt{0.01+25+0}=\sqrt{25.01}\approx 5.00$$

Distance from (6.5, 50, 4) → Fail

$$=\sqrt{(6.1-6.5)^2+(40-50)^2+(5-4)^2}$$
$$=\sqrt{0.16+100+1}=\sqrt{101.16}\approx 10.06$$

Distance from (5.8, 38, 5) → Fail

$$=\sqrt{(6.1-5.8)^2+(40-38)^2+(5-5)^2}$$
$$=\sqrt{0.09+4+0}=\sqrt{4.09}\approx 2.02$$

Distance	Label
2.02	Fail
5.00	Fail
10.06	Fail

Final Prediction Fail

5b)

1. **Simple Linear Regression**

- Models the relationship between a **single input variable** (X) and a **continuous output variable** (Y).
- The hypothesis function is linear:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

- Goal: Minimize the sum of squared errors between predicted and actual values.

---

2. **Multiple Linear Regression**

- Extends simple linear regression by using **multiple independent variables**:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + \epsilon$$

- Captures more complex relationships using multiple features.

---

3. **Polynomial Regression**

- Fits a **non-linear curve** by including polynomial terms of input features:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + ... + \beta_n X^n + \epsilon$$

- Useful when data shows **curvature**.

4. **Ridge Regression**

- Also called **L2 regularization**.
- Adds penalty term $\lambda \sum \beta_i^2$ to the cost function.
- Helps in **reducing model complexity and multicollinearity**.

---

5. **Lasso Regression**

- Also called **L1 regularization**.
- Adds penalty term $\lambda \sum |\beta_i|$ to the cost function.
- Performs **feature selection** by shrinking some coefficients to **zero**.

---

6. **Logistic Regression** *(used for classification)*

- Although called "regression," it is used for **binary classification** problems.
- Models the **probability** that a given input belongs to a class using the **logistic sigmoid function**:

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$$

Limitations of Regression Methods

(as discussed in Sridhar N textbook)

---

1. Assumption of Linearity

- Linear regression assumes that the relationship between input and output is **linear**.
- Fails when data exhibits **non-linear patterns**.

---

2. Sensitive to Outliers

- Regression models, especially least squares-based ones, are heavily influenced by **outliers**, which can skew results significantly.

---

3. Multicollinearity

- When independent variables are **highly correlated**, the model may become unstable and **coefficients may vary significantly**.

---

4. Overfitting

- Complex models like polynomial regression may fit the training data too well and fail to generalize on new data.

---

5. Irrelevant Features

- Including irrelevant or highly redundant features can **reduce model performance** unless techniques like **feature selection** or **regularization** are applied.

---

6. Poor Interpretability in High Dimensions

- In multiple regression with many variables, the **interpretation of coefficients becomes difficult**, especially with interactions or polynomial terms.

---

7. Limited to Numeric Input/Output

- Standard regression methods cannot handle **categorical outputs** or **textual input** unless preprocessed.

5c)

🔷 Step 1: **Check for Base Cases**

Before splitting:

1. **If all examples in D belong to the same class,**
   → return a **leaf node** with that class label.
2. **If the attribute list is empty,**
   → return a **leaf node** with the **majority class** in D.
3. **If the dataset is empty,**
   → return a **leaf node** with the **majority class** from the parent node's data.

🔷 Step 2: **Select the Best Attribute to Split**

- For each attribute, compute **Information Gain**:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

- Choose the attribute with **highest gain** as the **decision node**.

🔷 Step 3: **Create a Decision Node**

- Make the selected attribute the **root of the current subtree**.

🔷 Step 4: **Branch on Attribute Values**

- For each **value** v of the selected attribute:
  - Partition the dataset into a subset S_v where the attribute has value v.
  - Recursively **apply ID3** on S_v with the **remaining attributes**.

🔷 Step 5: **Repeat Until Base Case is Met**

- The recursion stops when:
  - All examples are of the same class
  - No attributes remain
  - Subset is empty

6a) Nearest centroid algorithm steps

✅ Step 2: Compute Class Centroids

🔷 Class A: (3,1), (5,2), (4,3)

$$\text{Centroid}_A = \left(\frac{3+5+4}{3}, \frac{1+2+3}{3}\right) = (4.0,\ 2.0)$$

🔷 Class B: (7,6), (6,7), (8,5)

$$\text{Centroid}_B = \left(\frac{7+6+8}{3}, \frac{6+7+5}{3}\right) = (7.0,\ 6.0)$$

✅ Step 3: Compute Euclidean Distances from Test Point (6,5)

📏 Distance to **Centroid A** (4.0, 2.0):

$$\sqrt{(6-4)^2 + (5-2)^2} = \sqrt{4+9} = \sqrt{13} \approx 3.61$$

📏 Distance to **Centroid B** (7.0, 6.0):

$$\sqrt{(6-7)^2 + (5-6)^2} = \sqrt{1+1} = \sqrt{2} \approx 1.41$$

CLASS =B

6b)

Regression	Correlation
Predicts the value of a dependent variable based on one or more independent variables	Measures the strength and direction of a linear relationship between two variables
Directional – predicts Y from X	Non-directional – just quantifies the relationship
Predictive equation (e.g., $Y = aX + b$)	A correlation coefficient (r), between -1 and +1
Used for prediction and modeling	Used to measure association
May suggest causation (with caution)	Does not imply causation

Regression	Causation
Shows mathematical relationship between variables	Implies that a change in one variable directly causes a change in another
Based on statistical data	Requires controlled experiments or strong assumptions
Can be used to predict outcomes	Explains why something happens
Increase in study time predicts higher marks	Studying causes improvement in marks

Linearity	Non-Linearity
Output is a linear combination of inputs	Output depends on non-linear transformations of inputs
Straight line in 2D	Curved, exponential, polynomial, etc.
$Y = aX + b$	$Y = aX^2 + bX + c$, $Y = a\sin(X) + b$, etc.
Easy to interpret	More complex to interpret
Linear regression	Polynomial regression, logistic regression, neural networks

6c) **Advantages:**

1. **Simple to Understand and Interpret**
   - Decision trees mimic human decision-making.
   - Easy to visualize and explain.
2. **No Need for Feature Scaling or Normalization**
   - Works with both numerical and categorical data directly.
3. **Handles Both Classification and Regression**
   - Can be used for predicting both discrete labels and continuous values.
4. **Performs Feature Selection**
   - Automatically selects the most significant attributes (via measures like Information Gain or Gini index).
5. **Works Well with Missing Values**
   - Can handle missing values to some extent during splitting.

6. **Non-parametric**
   - No assumptions about the distribution of data.

📉 Disadvantages:

1. **Overfitting**
   - Can create very deep trees that perfectly fit training data but fail on test data (low generalization).
2. **Instability**
   - Small changes in data can lead to a completely different tree (due to greedy splits).
3. **Biased Toward Features with Many Levels**
   - Attributes with many distinct values may be favored (especially with Information Gain).
4. **Less Accurate Than Ensemble Methods**
   - Individual trees are less powerful than Random Forest or Gradient Boosted Trees.
5. **Hard to Capture Linear Relationships**
   - Performs poorly if the true relationship between features and target is linear or additive

1. **If all examples in D belong to the same class**
   → Return a **leaf node** with that class label.
2. **If attribute set A is empty**
   → Return a **leaf node** with the **majority class** in D.
3. **Else:**
   - Select the **best attribute** A* from A using a splitting criterion (like Information Gain or Gini Index).
   - Create a **decision node** based on A*.
4. **For each value** v_i of attribute A*:
   - Partition the dataset:

   $$D_i = \{x \in D \mid A^* = v_i\}$$

   - **If** D_i **is empty**
     → Attach a **leaf node** with the **majority class** in D.
   - **Else**
     → Recursively call the algorithm on D_i and $A - \{A^*\}$
5. **Return the constructed tree**

7a)

✅ Step 2: Count Class Frequencies

Total examples: 10

- `Yes` → 4 instances
- `No` → 6 instances

So:

- $P(\text{Yes}) = 4/10 = 0.4$
- $P(\text{No}) = 6/10 = 0.6$

✅ Step 3: Count Conditional Probabilities

For class = Yes:

Attribute	Value	Count
Color	Red	3
Type	SUV	0
Origin	Domestic	2

So:

- $P(\text{Red}|\text{Yes}) = 3/4$
- $P(\text{SUV}|\text{Yes}) = 0/4 \rightarrow \text{Apply Laplace Smoothing: } (0+1)/(4+3) = 1/7$
- $P(\text{Domestic}|\text{Yes}) = 2/4 = 0.5$

For class = No:

Attribute	Value	Count
Color	Red	1
Type	SUV	3
Origin	Domestic	2

So:

- $P(\text{Red}|\text{No}) = 1/6$
- $P(\text{SUV}|\text{No}) = 3/6 = 0.5$
- $P(\text{Domestic}|\text{No}) = 2/6 \approx 0.333$

✅ Step 4: Compute Posterior Probabilities

For `Yes`:

$$P(\text{Yes}|\text{X}) \propto P(\text{Yes}) \times P(\text{Red}|\text{Yes}) \times P(\text{SUV}|\text{Yes}) \times P(\text{Domestic}|\text{Yes})$$

$$= 0.4 \times \frac{3}{4} \times \frac{1}{7} \times 0.5 = 0.4 \times 0.75 \times 0.1429 \times 0.5 \approx 0.0214$$

For `No`:

$$P(\text{No}|\text{X}) \propto P(\text{No}) \times P(\text{Red}|\text{No}) \times P(\text{SUV}|\text{No}) \times P(\text{Domestic}|\text{No})$$

$$= 0.6 \times \frac{1}{6} \times 0.5 \times 0.333 \approx 0.6 \times 0.1667 \times 0.5 \times 0.333 \approx 0.0167$$

✅ Step 5: Compare and Predict

- $P(\text{Yes}|\text{X}) \approx 0.0214$
- $P(\text{No}|\text{X}) \approx 0.0167$

$$\boxed{\text{Predicted class} = \text{Yes}}$$

7b)Diagrams+

Feature	Description
Input Layer	Accepts data inputs ($x_1, x_2, ..., x_n$)
Weights & Bias	Learnable parameters
Weighted Sum	Computes total input signal
Activation Function	Adds non-linearity to model
Output	Produces final neuron output

8a)

📘 1. Bayes' Theorem

🔷 Definition:

Bayes' Theorem is a fundamental concept in probability theory that describes how to **update the probability of a hypothesis** based on new evidence.

$$P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)}$$

Where:

- h: a hypothesis
- D: observed data (evidence)
- $P(h|D)$: **posterior probability** (probability of hypothesis h given data D)
- $P(D|h)$: **likelihood** (probability of data D given that h is true)
- $P(h)$: **prior probability** of h
- $P(D)$: **evidence** or marginal likelihood (constant across all hypotheses)

✅ 2. Maximum a Posteriori (MAP) Hypothesis

🔷 Definition:

The **MAP hypothesis** is the **most probable hypothesis** given the observed data, taking both the likelihood and the prior probability into account.

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

Using Bayes' Theorem:

$$h_{MAP} = \arg\max_{h \in H} \frac{P(D|h) \cdot P(h)}{P(D)} = \arg\max_{h \in H} P(D|h) \cdot P(h)$$

🔶 Key Point:

- MAP considers both the **data** (via $P(D|h)$) and **prior belief** (via $P(h)$)
- Useful when you have **prior knowledge** about hypotheses

✅ 3. Maximum Likelihood (ML) Hypothesis

🔷 Definition:

The **ML hypothesis** is the hypothesis that **maximizes the likelihood** of the observed data, **ignoring prior probabilities**.

$$h_{ML} = \arg\max_{h \in H} P(D|h)$$

🔶 Key Point:

- ML assumes **all hypotheses are equally likely a priori** (i.e., uniform prior)
- It is purely **data-driven**

8b)

✅ Activation Functions (as mentioned by Sridhar N)

Function	Expression	Range	Use Case
Step	$f(z) = 1 \text{ if } z \geq 0$	{0, 1}	Perceptron, binary classification
Sigmoid	$f(z) = \frac{1}{1+e^{-z}}$	(0, 1)	Logistic regression, smooth
Tanh	$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	(−1, 1)	Normalized output
ReLU	$f(z) = \max(0, z)$	[0, ∞)	Deep learning (hidden layers)

Explain in detail.

9a)

🔷 Step 1: Calculate Euclidean Distance to Centroids

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Obj	Point	Distance to (4,6)	Distance to (12,4)	Assigned Cluster
1	(2,4)	$\sqrt{(2-4)^2+(4-6)^2} = \sqrt{8} \approx 2.83$	$\sqrt{(2-12)^2+(4-4)^2} = \sqrt{100} = 10.0$	C1
2	(4,6)	$\sqrt{0} = 0.0$	$\sqrt{(4-12)^2+(6-4)^2} = \sqrt{68} \approx 8.25$	C1
3	(6,8)	$\sqrt{(6-4)^2+(8-6)^2} = \sqrt{8} \approx 2.83$	$\sqrt{(6-12)^2+(8-4)^2} = \sqrt{52} \approx 7.21$	C1
4	(10,4)	$\sqrt{(10-4)^2+(4-6)^2} = \sqrt{40} \approx 6.32$	$\sqrt{(10-12)^2+(4-4)^2} = \sqrt{4} = 2.0$	C2
5	(12,4)	$\sqrt{(12-4)^2+(4-6)^2} = \sqrt{68} \approx 8.25$	$\sqrt{0} = 0.0$	C2

✅ Step 2: New Clusters after Assignment

- Cluster C1: (2,4), (4,6), (6,8)
- Cluster C2: (10,4), (12,4)

🔷 Step 3: Compute New Centroids

📍 New Centroid of C1:

$$x = \frac{2+4+6}{3} = 4.0, \quad y = \frac{4+6+8}{3} = 6.0 \rightarrow (4.0,\ 6.0)$$

📍 New Centroid of C2:

$$x = \frac{10+12}{2} = 11.0, \quad y = \frac{4+4}{2} = 4.0 \rightarrow (11.0,\ 4.0)$$

✅ Step 4: Check for Convergence

- Old centroids: (4,6), (12,4)
- New centroids: (4,6), (11,4)

9b)

Explain each component in detail

Component	Role
Agent	Learns and makes decisions
Environment	Responds to agent's actions
State (S)	Current situation
Action (A)	Options available to the agent
Reward (R)	Feedback signal
Policy (π)	Decision-making strategy
Value (V)	Expected long-term reward
Q-Function	Expected reward for a state-action pair
Model	Predicts next state/reward (used in model-based RL)

10a)

📏 1. **Manhattan Distance**

Also called L_1 **norm** or **city block distance.**

$$D_{\text{Manhattan}} = |x_2 - x_1| + |y_2 - y_1|$$

$$= |5 - 0| + |8 - 3| = 5 + 5 = \boxed{10}$$

📏 2. **Chebyshev Distance**

Also called $L\infty$ **norm** or **maximum metric.**

$$D_{\text{Chebyshev}} = \max(|x_2 - x_1|,\ |y_2 - y_1|)$$

$$= \max(|5 - 0|,\ |8 - 3|) = \max(5,\ 5) = \boxed{5}$$

10b)

1. Choose kernel & bandwidth
2. For each point:
   - Find neighbors within radius
   - Compute their mean
   - Shift point toward the mean
3. Repeat until convergence
4. Assign clusters based on convergence points

Explain each step

10c) 🔷 **Characteristics of Reinforcement Learning**

1. **Trial-and-Error Learning**
   - The agent **learns by interacting** with the environment and improving its actions based on feedback.
2. **Delayed Reward**
   - Actions may not have an **immediate impact**, and rewards can be delayed across multiple time steps.
3. **Exploration vs Exploitation**
   - The agent must **explore new actions** to discover better rewards but also **exploit known actions** that yield high rewards.
4. **Sequential Decision Making**
   - Decisions affect **future states and rewards**, forming a sequence of learning episodes.
5. **Feedback-Based Learning**
   - Learning is guided only by a **scalar reward signal**, not labeled input-output pairs (unlike supervised learning).
6. **Learning Optimal Policy**
   - The goal is to learn a **policy** π(a|s)\pi(a|s)π(a|s) that maximizes the expected cumulative reward.

🔷 Challenges in Reinforcement Learning

1. **Exploration–Exploitation Dilemma**
   - Balancing **trying new actions** vs **choosing the best known action** is non-trivial.
2. **Delayed and Sparse Rewards**
   - Sometimes rewards come only after a long sequence of actions (e.g., playing a game), making learning difficult.
3. **Credit Assignment Problem**
   - Determining **which action** was responsible for a reward is hard when many actions contributed.
4. **Large/Continuous State Spaces**
   - When the environment has large or continuous state/action spaces, it becomes **computationally challenging**.
5. **Non-Stationary Environments**
   - The environment might **change over time**, requiring adaptive learning.
6. **Partial Observability**
   - The agent might **not fully observe** the state of the environment, making learning incomplete or uncertain.
7. **Function Approximation**
   - Using models like **neural networks** to approximate value functions can lead to **instability**.

🔷 Applications of Reinforcement Learning

Domain	Application Example
Games	Playing Chess, Go, Atari, AlphaZero, Dota 2
Robotics	Motion planning, robotic arm control

Domain	Application Example
AI & Planning	Self-learning agents, intelligent planning
Simulation	Game AI agents, navigation tasks
Network Systems	Routing optimization, congestion control
Industrial Control	Adaptive controllers, process optimization
Healthcare	Treatment planning, drug dosage adjustment
Finance	Stock trading, portfolio optimization
Autonomous Systems	Drone path planning, self-driving cars